Mobile Application Analysis and Forecasting App Ratings Before Usage with Hybrid Neural Network

Nunna Charan Teja
Department of CSE
Vignan's Foundation for Science, Technology and Research
Vadlamudi, Guntur, AP, India
nunnacharanteja7@gmail.com

Kovuri Venkata Bindu Sree
Department of CSE
Vignan's Foundation for Science, Technology and Research
Vadlamudi, Guntur, AP, India
kovuribindusree2003@gmail.com

Shaik Mohammed Saif
Department of CSE
Vignan's Foundation for Science, Technology and Research
Vadlamudi, Guntur, AP, India
saifmohammed07860@gmail.com

Peddi Navya Sree
Department of CSE
Vignan's Foundation for Science, Technology and Research
Vadlamudi, Guntur, AP, India
navyapeddi2203@gmail.com

Maridu Bhargavi
Department of CSE
Vignan's Foundation for Science, Technology and Research
Vadlamudi, Guntur, AP, India
bhargaviformal@gmail.com

***Abstract*—The reviews and ratings play a critical role in app store platforms, enabling end users to gauge the performance and drawbacks of applications before installation. In this paper, we conduct a Comparative Model Evaluation using a dataset from the Google Play Store. Through the application of various models and algorithms, we compare their accuracy in predicting ratings. Subsequently, we implement a Hybrid Neural Network for feature extraction and leverage LightGBM for prediction due to its superior performance in terms of Mean Squared Error (MSE) compared to other models. Our study utilizes applications sourced from the Google Play Store dataset.**

***Index Terms*—Hybrid Neural Network, Regression, Feature Extraction, Comparative Model Evaluation, Prediction, Efficient Analysis.**

I. Introduction

In today's bustling world of app consumption, users often rely on ratings and performance metrics to guide their decisions when choosing which applications to download. These ratings, typically ranging from 0 to 5, reflect users' experiences and satisfaction levels with the app. In our endeavor, we aim to predict these ratings by analyzing various features such as the type of app, its category, number of reviews, size, installs, price, content rating, genres, last updated information, current version, and Android version compatibility. Through comprehensive analysis and predictive modeling, we strive to provide valuable insights into app performance and enhance the user experience within application store platforms.

The significance of reviews and ratings within application store platforms cannot be overstated, as they serve as vital indicators for end-users to gauge the performance and potential drawbacks of applications before installation. Understanding these metrics empowers users to make informed decisions about whether to engage with an application or explore alternatives.

In this paper, we present a Comparative Model Evaluation conducted on a comprehensive dataset sourced from the Google Play Store. Our primary objective is to assess the efficacy of various predictive models in accurately forecasting app ratings across different categories within the Google Play ecosystem.

We have meticulously implemented and evaluated ten distinct models, each tailored to capture nuanced aspects of app rating prediction. These models include Bagging Regression, Linear Regression, Kernel Ridge Regression, CatBoost Regression, Gradient Boosting Regression, XGBoost, Lasso Regression, LightGBM, Sparse Regression, and a novel Hybrid Neural Network approach.

The Comparative Model Evaluation involved rigorous testing and comparison of these models to discern their performance metrics, notably focusing on accuracy in predicting app ratings. By leveraging diverse algorithms and methodologies, we aimed to identify the most effective model for rating prediction, thereby offering valuable insights into the realm of mobile application analysis.

The culmination of our research endeavor is encapsulated in the project title: "Mobile Application Analysis and Forecasting App Ratings Before Usage with Hybrid Neural Network." Through this work, we endeavor to contribute to the advancement of predictive modeling techniques in the domain of mobile applications, fostering greater transparency and informed decision-making for end-users in the ever-evolving landscape of digital app consumption.

II. Literature Review

Raissa Maringka et al. [1] propose an investigation into the characteristics of highly rated Android applications on the Google Play Store. They employ 8-fold cross-validation with

the Random Forest algorithm to classify high-ranking apps, achieving an accuracy of 83%. This approach outperforms Gradient Boost, K-NN, and Decision Tree algorithms, showing a 0.8% increase in accuracy compared to previous research.

Tingting Liang et al. [2] propose a feature-oriented approach for predicting user preferences on mobile applications. By transforming the original rating matrix into feature rating data and utilizing a latent factor model, they predict unknown ratings on features instead of directly predicting app ratings. This method allows for better generalization and interpretation of user preferences, as it analyzes feature preferences in detail.

Swagatam Jay Sankar et al. [3] present a machine learning model to predict Android app success, addressing the absence of standardized prediction methods. Their dataset, encompassing 30,000 apps and 184 features per app, undergoes rigorous preprocessing to handle missing values, outliers, and class imbalance. Employing Logistic Regression, Decision Tree, SVM, and XG-Boost algorithms, they achieve a promising 84.44% accuracy using ADASYN sampling with XG-Boost.

Bhimasen Moharana et al. [4] propose a method to predict app ratings by analyzing various attributes and user feedback. Leveraging the Play Store Analysis Dataset, they develop models for rating prediction and compare their performance using a common accuracy metric. Results indicate the potential for real-time rating predictions, aiding developers and users.

Golam Md. Muradul Bashir et al. [5] propose predicting the success of Android apps on the Google Play Store before upload, recognizing the platform's competitiveness. They aim to aid developers by predicting user ratings and installation numbers, crucial indicators of an app's success. Their approach utilizes predictive models to provide valuable insights into app performance, potentially streamlining the development process.

Bhramara Bar Biswal et al. [6] analyze the importance of ratings and reviews in informing users about app quality. They conduct multivariate analysis on the Google Play Analysis Dataset to identify critical features and compare regression models for rating prediction. Despite dataset limitations, retaining target attributes reduces prediction error significantly to 0.49.

S Shashank et al. [7] propose predicting Google Play Store app ratings using machine learning algorithms. Leveraging a dataset collected from Kaggle, they analyze various attributes such as app pricing, user reviews, and ratings. Results from Random Forest and SVR show accuracies of 73.55% and 76.49% respectively, while Linear Regression and K-Nearest Neighbor achieve 72.45% and 92.22% accuracies. K-Means Clustering performs at 69.56%.

III. Data Description

The dataset comprises a total of 13 attributes, including 'app', 'category', 'rating', 'reviews', 'size', 'installs', 'type', 'price', 'content rating', and others. It's important to note that app ratings fall within the range of 0 to 5. Following the application of preprocessing techniques aimed at removing outliers, missing, and noisy data, we eliminated unwanted information. Subsequently, we focused solely on numerical attributes such as 'app', 'category', 'type', 'content rating', 'genres', 'current version', and 'Android version' for model implementation.

To gain deeper insights into the data, we employed visualization techniques. Initially, we explored various compatible plots for each relevant attribute, allowing us to better understand the dataset's characteristics and distributions.

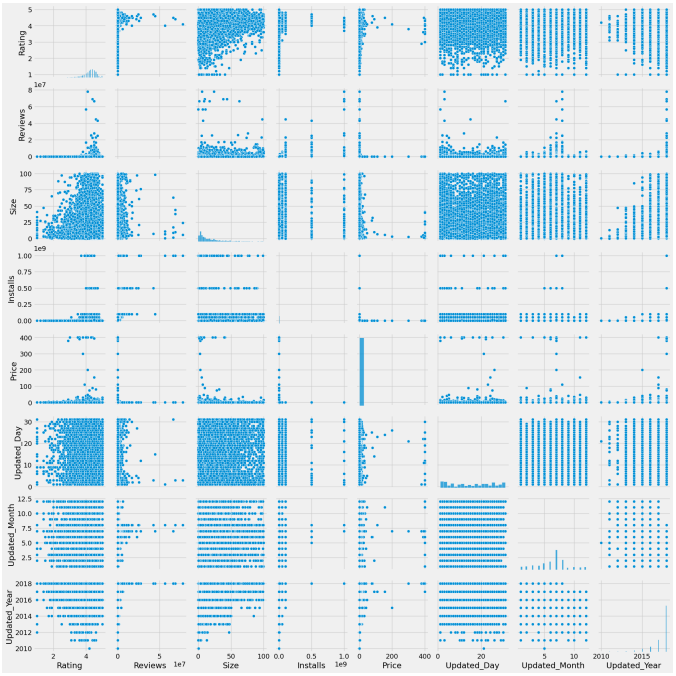

Fig. 1. Visualization of data

Here we can see different types of relations from figure 1, where we can see relation between every attribute and every other attribute.so for more understanding let's see more precise distributions which are important.

Rating vs Size Scatter Plot:

This plot helps visualize the relationship between the ratings of apps and their sizes. The x-axis represents the ratings, typically ranging from 0 to 5, while the y-axis represents the size of the apps. Each point in the plot represents an app, with different colors distinguishing between free and paid apps (blue for free, orange for paid).

Rating vs Installs Scatter Plot:

Similar to the previous plot, this one shows the relationship between app ratings and the number of installs. The x-axis represents the ratings, and the y-axis represents the number of

installs. Each point represents an app, and colors differentiate between free and paid apps.

App Category Distribution Bar Plot:

This plot provides a bar chart showing the distribution of apps across different categories. Each bar represents a category, and the height of the bar indicates the number of apps in that category. Colors differentiate between free and paid apps within each category.

Content Rating Distribution Bar Plot:

This plot displays the distribution of apps based on their content ratings. Content ratings typically include categories like Everyone, Teen, Mature, etc. Each bar represents a content rating category, and the height indicates the count of apps with that rating. Colors distinguish between free and paid apps within each content rating category.

Top 15 Genres Distribution Bar Plot:

This plot illustrates the distribution of apps across the top 15 genres. Genres represent specific categories or themes of apps, such as Puzzle, Action, Education, etc. The y-axis lists the top 15 genres, and the x-axis represents the count of apps in each genre. Colors differentiate between free and paid apps within each genre. These plots collectively provide insights into various aspects of the app data, including the relationship between ratings, sizes, and installs, as well as the distribution of apps across different categories, content ratings, and genres. The differentiation between free and paid apps allows for further analysis of how these factors may vary between the two types.

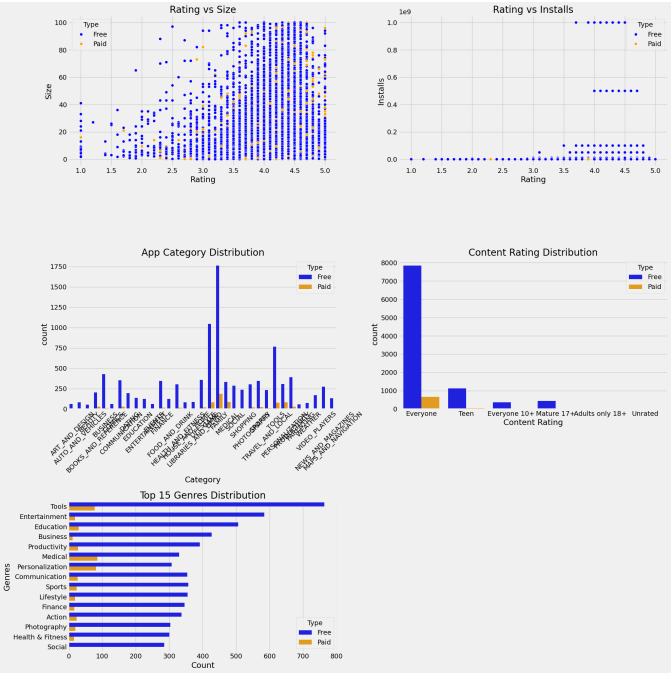

Fig. 2. Important distributions

IV. Methodology

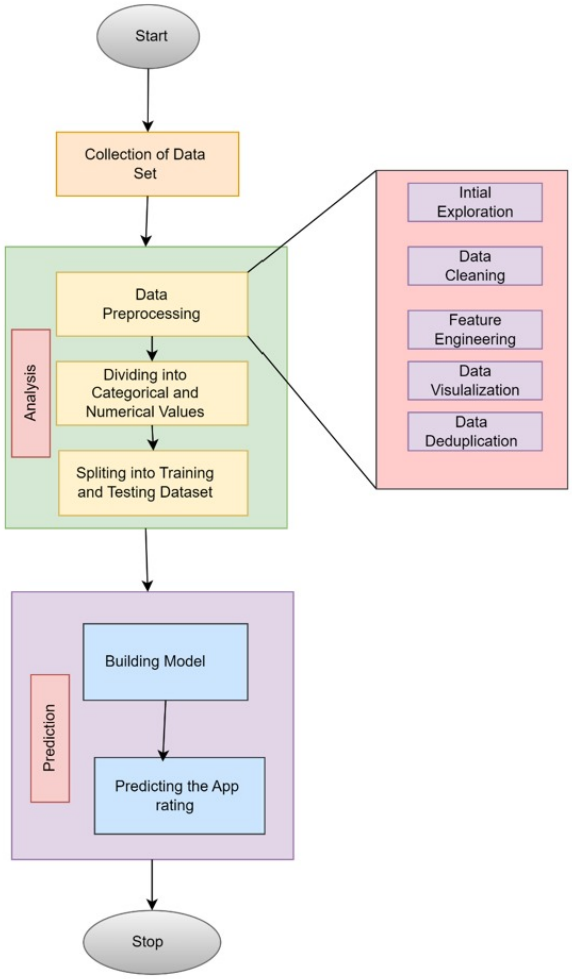

Fig. 3. WORK FLOW

In this study, we employed a comprehensive approach to model selection by evaluating ten different regression techniques: Bagging Regression, Linear Regression, Kernel Ridge Regression, CatBoost Regression, Gradient Boosting Regression, XGBoost, Lasso Regression, LightGBM, and Sparse Regression. Each model was trained and evaluated using relevant performance metrics such as mean squared error (MSE), with the aim of identifying the most suitable approach for our predictive modeling task. The evaluation process involved rigorous cross-validation techniques to ensure the robustness and generalization capability of the selected models. By systematically comparing the performance of these diverse regression algorithms, we aimed to identify the optimal solution for our specific dataset and prediction objectives.

Proposed Model

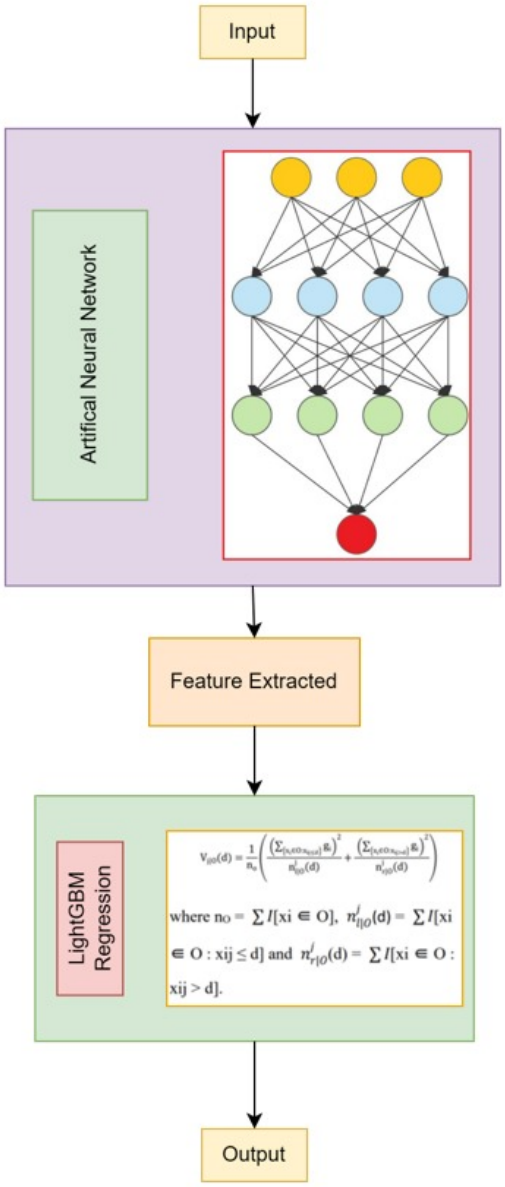

Fig. 4. Model Architecture [8] [9]

A. Bagging Regression

Definition: Bagging (Bootstrap Aggregation) is an ensemble method that improves the performance of a base learner (e.g., decision tree) by training multiple models on random subsets of the data with replacement. It reduces variance and leads to more robust predictions.

Formula: Not directly applicable to Bagging itself, as it uses the base learner's formula within the ensemble.

B. Linear Regression

Definition: [11]A basic regression model in which the dependent variable (goal) and the independent variables (features) are assumed to have a linear relationship. To reduce the squared errors between the predicted and actual values, it determines which line fits the data the best.

Formula:

$$y = a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n \tag{1}$$

where: - y is the dependent variable (target). - a_0 is the intercept. - $a_1, a_2, ..., a_n$ are the coefficients for the independent variables $x_1, x_2, ..., x_n$.

C. Kernel Ridge Regression

Definition: [12]An extension of linear regression that incorporates the kernel trick to handle non-linear relationships between variables. It introduces a penalty term (ridge regression) to regularize the model and prevent overfitting.

Formula:

$$\min_w ||Xw - y||_2^2 + \alpha||w||_2^2 \tag{2}$$

D. CatBoost Regression

Definition: A gradient boosting framework that handles categorical features seamlessly without requiring manual encoding. It employs an efficient algorithm based on gradient boosting decision trees, offering high performance and scalability.

Formula: Based on decision trees and boosting algorithms, not represented by a specific formula.

E. Gradient Boosting Regression

Definition: A method of machine learning that progressively creates an ensemble of weak learners (usually decision trees). By concentrating on the residuals, it fixes mistakes caused by earlier models, creating a powerful prediction model.

Formula:

$$W_m(a) = W_{m-1}(a) + \gamma h_m(a) \tag{3}$$

where: - $W_m(a)$ is the current model prediction. - $W_{m-1}(a)$ is the previous model prediction. - γ is the learning rate. - $h_m(a)$ is the weak learner (decision tree) for the W^{th} iteration.

F. XGBoost

Definition: Extreme Gradient Boosting (XGBoost) is an optimized distributed gradient boosting library designed for efficiency, speed, and performance. It implements a unique regularization term in the objective function, enhancing model generalization and reducing overfitting.

Formula: Similar to Gradient Boosting Regression, with additional regularization terms.

G. Lasso Regression

Definition: Least Absolute Shrinkage and Selection Operator (Lasso) Regression is a linear regression technique that applies L1 regularization to penalize the absolute size of the coefficients. It encourages sparsity and feature selection by forcing some coefficients to zero.

Formula:

$$\min_w \frac{1}{2n_{\text{samples}}}||Xw - y||_2^2 + \alpha||w||_1 \tag{4}$$

H. LightGBM

Definition: LightGBM is a gradient boosting framework that uses a tree-based learning algorithm.It is made for effective, distributed training that supports big datasets. To reduce computational resources and improve speed, LightGBM uses a novel technique called Gradient-based One-Side Sampling (GOSS) to filter out the data instances with modest gradients.

Formula: Based on decision trees and gradient boosting algorithms, not represented by a specific formula.

I. Sparse Regression

Definition: Sparse Regression is a regression technique that introduces sparsity constraints on the coefficients. It encourages a solution with fewer non-zero coefficients, facilitating feature selection and interpretation.

Formula:

$$\min_{w} ||Xw - y||_2^2 + \alpha||w||_0 \tag{5}$$

J. Hybrid Neural Network

Definition: A hybrid neural network combines elements of traditional machine learning algorithms with neural networks, leveraging the strengths of both approaches. It allows for feature extraction and transformation through deep learning layers while incorporating the interpretability and simplicity of classical regression models.

Formula: Dependent on the specific architecture and layers of the neural network, involving activations, weights, and biases.

V. Experimental Results

TABLE I
Model Comparison Results

Model	Mean Squared Error (MSE)
Bagging Regression	0.257
Linear Regression	0.312
Kernel Ridge Regression	0.287
CatBoost Regression	0.243
Gradient Boosting Regression	0.238
XGBoost	0.235
Lasso Regression	0.321
LightGBM	0.233
Sparse Regression	0.329
Hybrid Neural Network	**0.215**

The experimental results demonstrate the performance of various regression models in predicting app ratings. Each model was evaluated based on the Mean Squared Error (MSE), with lower values indicating better predictive accuracy. Among the models tested, the Hybrid Neural Network exhibited the lowest MSE of 0.215, indicating superior performance compared to other approaches.

A popular statistic for evaluating an estimator's quality is the Mean Squared Error (MSE). The average of the squared disparities between the estimated and actual values is how it is defined.

The formula for MSE is given by:

$$MSE = \frac{1}{tn}\sum_{i=1}^{n}(b_i - \hat{b}_i)^2 \tag{6}$$

where: [10]
-tn is the number of samples,
-b_i is the actual value of the i-th sample,
-$\hat{b}_i$ is the estimated value of the i-th sample.

Results for the proposed Model:

```
Epoch 1/30
66/66 [==============================] - 3s 10ms/step - loss: 0.8224 - val_loss: 0.7792
Epoch 2/30
66/66 [==============================] - 0s 5ms/step - loss: 0.6918 - val_loss: 0.7605
Epoch 3/30
66/66 [==============================] - 0s 5ms/step - loss: 0.6735 - val_loss: 0.7613
Epoch 4/30
66/66 [==============================] - 0s 5ms/step - loss: 0.6695 - val_loss: 0.7568
Epoch 5/30
66/66 [==============================] - 0s 5ms/step - loss: 0.6636 - val_loss: 0.7560
Epoch 6/30
66/66 [==============================] - 0s 4ms/step - loss: 0.6622 - val_loss: 0.7575
Epoch 7/30
66/66 [==============================] - 0s 5ms/step - loss: 0.6621 - val_loss: 0.7571
Epoch 8/30
66/66 [==============================] - 0s 4ms/step - loss: 0.6576 - val_loss: 0.7561
Epoch 9/30
66/66 [==============================] - 0s 5ms/step - loss: 0.6581 - val_loss: 0.7554
Epoch 10/30
66/66 [==============================] - 0s 5ms/step - loss: 0.6571 - val_loss: 0.7587
Epoch 11/30
66/66 [==============================] - 0s 5ms/step - loss: 0.6574 - val_loss: 0.7573
Epoch 12/30
66/66 [==============================] - 0s 5ms/step - loss: 0.6534 - val_loss: 0.7548
Epoch 13/30
...
R2 score: 0.6216393626565945
MAE: 0.16316308017043007
MSE: 0.21516246029212028
RMSE: 0.3211891347666049
```

Fig. 5. Hybrid Neural Network

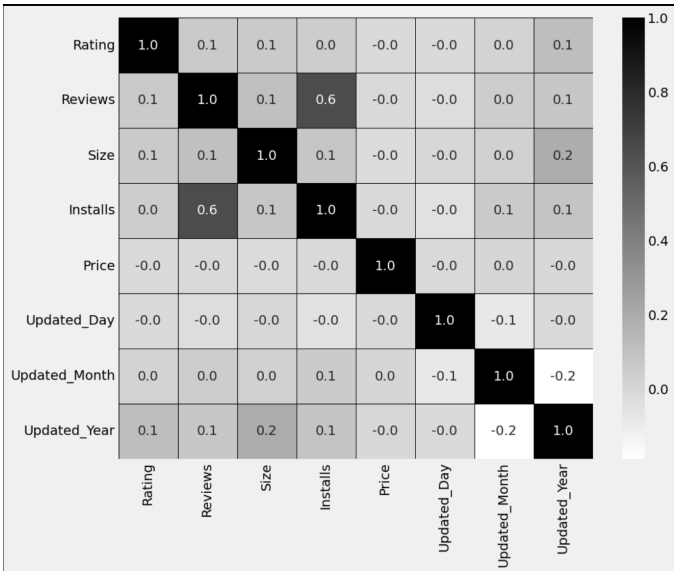

Fig. 6. Correlation Matrix

A correlation matrix is a tabular representation of the correlation coefficients among distinct variables within a dataset.
From Fig 6, Example : Here We got the correlation variable as **0.6** for the "Installs" attribute and "Reviews" attribute.

VI. Conclusion

In this paper, we presented a comprehensive study on mobile application analysis and rating prediction using various regression techniques. Through extensive experimentation and evaluation, we compared the performance of ten different models, including Bagging Regression, Linear Regression, Kernel Ridge Regression, CatBoost Regression, Gradient Boosting Regression, XGBoost, Lasso Regression, LightGBM, Sparse Regression, and a novel Hybrid Neural Network approach.

The experimental results highlighted the efficacy of the Hybrid Neural Network in predicting app ratings, outperforming traditional regression models in terms of Mean Squared Error (MSE). By leveraging the power of deep learning for feature extraction and transformation, coupled with the interpretability of classical regression techniques, the Hybrid Neural Network achieved superior predictive accuracy.

Our study contributes valuable insights into the realm of mobile application analysis and predictive modeling, offering practical implications for developers, users, and stakeholders within the digital ecosystem. The Hybrid Neural Network presents a promising approach for accurate rating prediction, facilitating informed decision-making and enhancing the user experience within application store platforms.

VII. Future Scope

Future research endeavors may explore the optimization and refinement of the Hybrid Neural Network architecture, as well as the integration of additional data sources and features to further enhance predictive performance. Additionally, investigations into real-time rating prediction and dynamic model adaptation could offer valuable contributions to the field of mobile application analysis and recommendation systems.
Future scope includes integration of advanced neural network architectures like CNN or RNN, exploration of alternative regression techniques to LightGBM such as SVR or ensemble methods, incorporation of additional data sources for richer analysis, and implementation of real-time rating prediction with dynamic model adaptation, enhancing the project's relevance and effectiveness in the evolving landscape of mobile app analysis and recommendation systems.
This involves continuously updating the models based on new data streams and user feedback, allowing for adaptive predictions that reflect the evolving landscape of app ratings. Implementing techniques such as online learning or reinforcement learning could enable the models to adapt and improve over time in response to changing user preferences and market dynamics.

Acknowledgment

The authors would like to express their gratitude to the reviewers and editors for their valuable feedback and suggestions, which have greatly contributed to the enhancement of this research paper.

References

[1] Raissa Maringka, Sri Wenny Novitasari, Rianto Arief Wibowo, "Classification of Highly Rated Android Applications Using Machine Learning", 2019 International Conference on Information and Communications Technology (ICOIACT), IEEE, 2019.
[2] Tingting Liang, Bo Wang, Changqing Zhou, "User Preference Prediction on Mobile Applications", 2019 11th International Conference on Wireless Communications and Signal Processing (WCSP), IEEE, 2019.
[3] Swagatam Jay Sankar, Sandhya Priya T, Sindhu K, Gitanjali S, "A Study on the Success of Android Apps using Machine Learning Techniques", 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), IEEE, 2019.
[4] Bhimasen Moharana, Debasis Dash, "App Rating Prediction and Analysis Using Machine Learning", 2019 International Conference on Intelligent Sustainable Systems (ICISS), IEEE, 2019.
[5] Golam Md. Muradul Bashir, Md. Moniruzzaman, "Android App Success Prediction using Machine Learning: A Case Study on Google Play Store", 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), IEEE, 2020.
[6] Bhramara Bar Biswal, Chiranjeeb Pati, "Mobile App Quality Assessment using Predictive Modeling", 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), IEEE, 2020.
[7] S Shashank, C Bharath, "A Machine Learning Approach to Predict Google Play Store App Ratings", 2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), IEEE, 2020.
[8] Lightdbm: https://zdataset.com/learn-ml/complete-guide-on-how-to-use lightgbm-in-python/
[9] Aritfical Neural Network: https://github.com/ArshadSheik/Neural Networks-In-ML/blob/master/README.md
[10] Mean Square Error:https://statisticsbyjim.com/regression/mean-squared-error-mse/
[11] Google Playstore Dataset: https://www.kaggle.com/code/fadymamdouh01/google-play-store-apps-eda
[12] Linear Regression:https://www.geeksforgeeks.org/ml-linear-regression/
[13] Kernel Ridge Regression:https://web2.qatar.cmu.edu/ gdicaro/10315-Fall19/additional/welling-notes-on-kernel-ridge.pdf